polka - タグとリンクでつながる新しい分散Web

開発駆動コース 仲山ゼミ
安田陽真

開発の動機

ある日、Twitterのアカウントが誤BANされました。自分のデータ、コミュニティ、いろいろなものに突然アクセスできなくなりました。幸い、数週間でアカウントは復活しましたが、「自分のデータであるはずなのに、自分ではコントロールできない。」 Googleアカウントによってアイデンティティを証明し、投稿やつながり、履歴のすべてをサービスのクラウドに預けていることの脆弱さを、このとき初めて実感しました。純粋なWebという最も普及していて分散したインフラを活用して、自分のデータをコントロール/検証できないか、と考えたことがpolkaを開発したきっかけです。

既存Web(Web 2.0)の課題

アカウントBAN

プラットフォームは、ユーザーのアカウントを企業内の判断のみで凍結、削除することができます。異議申し立て手段が用意されているプラットフォームもありますが、裁判のように慎重な手続きを踏んで行われるものでもありません。

サービス終了

ユーザーデータは、すべて運営側が管理しているサーバーに集中します。運営がそのインフラを維持できなくなれば、ユーザーのデータはすべて失われてしまいます。ユーザー側の対応手段もありません。

API制限

Web 2.0的な世界では、サービスはAPIとしてデータを公開することで、サービス同士が相互作用をすることができます。しかし、APIとしてデータを提供するかどうかはサービス側にゆだねられており、有料化などでアクセスが制限される可能性もあります。

これらの問題は、データとサービスが一体化していることで起きている！

Web 2.0的な世界では、データはサービスと密接に結びつき、切っても切れないものになっています。この部分を切り離し、データはユーザーが公開、**サービスは集計 or 表示担当**とすることで、これらの課題を解決できないかと考えました。

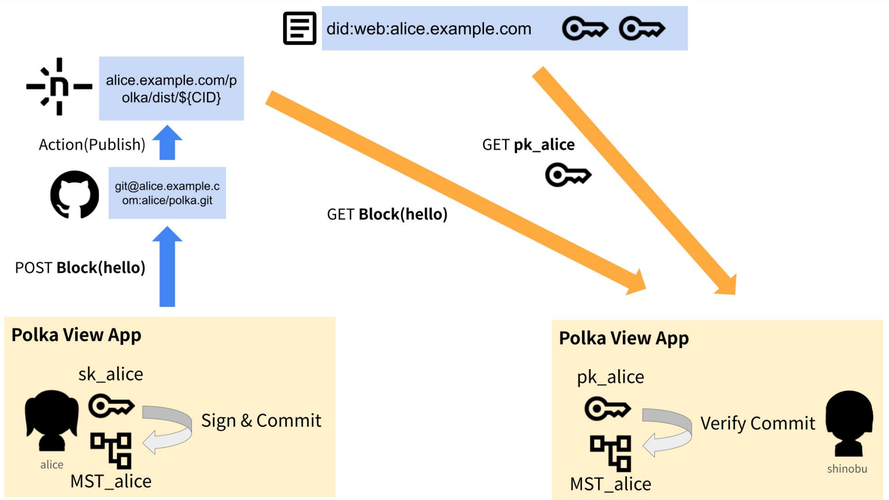
Webというインフラ

自分のマシンでネットワークにファイルを公開して、それが相互にリンクされていく、そんな基本のWebは、非常に分散していて、シンプルなものでした。この基本のWebインフラを使って、ソーシャル的な相互作用と検証可能性を実現できないでしょうか?

集計と分散のトレードオフ

Webにちらばった無数のデータを集めるためには、中央サーバーを用意するのが高速です。分散すればするほど、グローバルな情報を集計することが難しくなります。ですが、必ずしもすべてのWeb世界がグローバルな情報を瞬時に受信できるものである必要があるのでしょうか?

必要なのは、Webに散らばったデータを賢くリンクするしくみ



左の図は、polkaの全体像を示しています
aliceは自分の秘密鍵でMSTのルートに署名をし、コミットをGit RemoteにPushします。
コミットはサーバーにPublishされます。
shinobuはaliceのDIDをresolveしてaliceの公開鍵を取得、MSTとルートを取得して検証します。

polkaのアプローチ

データとサービスの分離

polkaはデータとサービスを分離します。データはユーザーの手元に配置し、サービスはそれを集計または表示するだけの存在と位置付けます。

検証可能性

polkaでのすべてのアクティビティは検証可能です。ユーザーは自分の手元に検証可能なデータベースを配置し、そこにデータを書き込みます。

リンク構造/フォロー

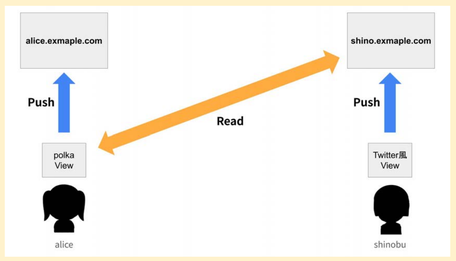
polkaは、各ユーザーが階層的なトピック構造を持ち、トピックごとのフォローや、投稿のリンクなどが可能です。

ユーザー発見

同じタグを持つユーザー同士は、自然と発見しあうことができます。このユーザー発見を、だれでもホストできるリレー上で行っています。

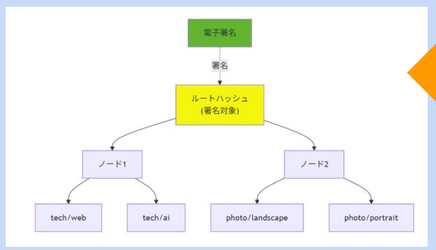
polkaを使おう!

データの流れ



アプリケーションで使用するデータは、すべてユーザーが指定したGitリポジトリに保存されます。aliceがshinobuの投稿を取得する際は、shino.example.comに直接リクエストを投げます

検証可能性



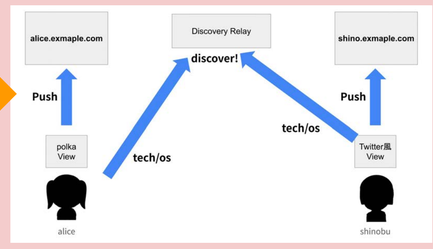
polkaの検証可能性を実現するためには、投稿本体だけでなく、投稿の階層構造のすべてが一度に署名、検証できる必要があります。そこで、ATProtocolのリポジトリ構造を採用しています。ATProtocolのリポジトリは、Merkle Search Treeという検証可能なハッシュ木を利用することで、大量のデータを一度に検証することが可能です。また、Merkle Search Treeは範囲に基づくQueryを処理できます。

階層構造

ユーザーの投稿は、ファイルシステムのような階層構造になっています。このような構造にすることで、階層的な購読を行うことができます。例えば、aliceのtech/webをフォローすることでwebフォルダ以下の投稿を、tech/をフォローすることでtechフォルダ以下のすべての投稿をフォローすることができます。

投稿ビューワーと
タグ階層グラフ

ユーザー発見



Nostrプロトコルを利用してユーザー発見を行います。
自分のタグ情報(Bloom Filter)を広告、
リレーをサブスクライブすることで同じタグを
グラフに持つユーザーをマッチングします。

Nostrイベントの例

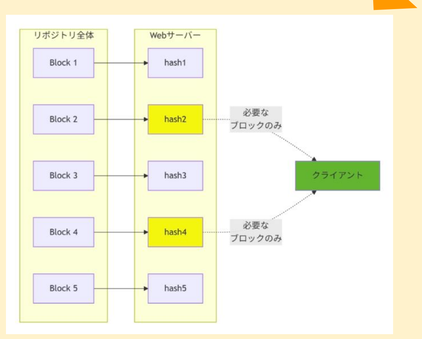
セットアップCLI

polkaには、セットアップCLI/鍵管理TUIがあります。秘密鍵はage + 共通鍵でローカルに保存され、TUIから管理することができます。リポジトリの初期化などのセットアップはCLIから行うことができます。

デスクトップアプリ

polkaのViewアプリはElectronベースのデスクトップアプリです。フロントエンドはSolid.jsで構築されていて、IPC経由でバックエンドと通信します。
フロントエンドのCSSフレームワークにはPico CSSを採用していて、フロントエンドのVibe Codingの安定性を高めています。

Blockstore



ATProtocolのリポジトリを静的ファイルとして公開するために、Blockstoreという仕組みを使用しています。Blockstoreは、CID(ファイルのハッシュ)をkey、データをvalueとするデータ構造で、polkaではこのブロック単位でサーバーに静的ファイルとして配置、クライアント側でMSTをたどりながらその都度必要なブロックを取得しています。

did:web

polkaでは、did:webをアイデンティティの基盤として利用しています。did:webを使用することで、ユーザーは自分のサーバーにJSONファイルを配置するだけでシンプルに利用を開始することができ、複数の鍵をローテーションすることも可能になります。

検証への考え方

polkaでは、「連続したアイデンティティとしての検証」と「暗号学的な検証」の二つを基盤としています。 暗号学的な検証は、did:webおよびMSTへの署名を通じて実施されます。これにより、ユーザーとの関係性の中で形成されるアイデンティティを検証する際、公開鍵の一致などを補助的な根拠として利用できます。

フォロー設計

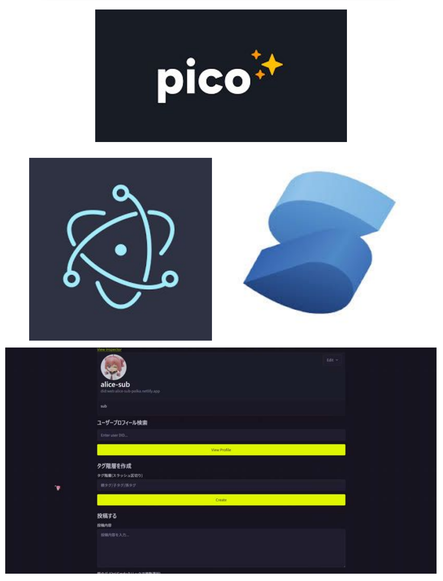
ユーザーをフォローしたら、自分の想定よりもたくさんのテーマの投稿が流れてきた経験はありませんか?polkaのフォローのコアとなる概念は、ユーザーではなく、タグをフォローすることです。階層構造の例のように、すべての投稿を取得するのではなく、特定のカテゴリの投稿のみを取得することができます。これにより、より自分で得たい情報をコントロールすることができます。また、polkaではタグをフォローしたことは相手に通知されません。あえてフォローを通知しないことで、フォローをしてもらったからには返さなくてはいけない、のような心理を働きにくくします。これはあくまでフォローは自分の取得する情報を選別するための「購読」であるという設計思想に基づいています。

テーマごとのインスタンスとは何が違う?

MisskeyやMastodonには、テーマごとのインスタンスが存在し、同じような興味を持った人たちが所属して交流します。このような専用インスタンスには、同じ興味を持つ人と手軽に交流できるというメリットがありますが、実際にインスタンスをのぞいてみると、そのインスタンスに限らず様々なインスタンスをまたがって同じ興味で交流しています。このことから私は、コミュニティはインスタンスではなく興味で自然発生すると考えています。では、インスタンスという「所属」を固定する必要はあるのでしょうか?polkaはそのような考えから専用インスタンスの設計をとっていません。

Bloom Filterの使用

polkaのタグ広告では、確率的データ構造であるBloom Filterを使用しています。Bloom Filterを利用することで、直接的なタグ情報をリレーに公開せずにタグが含まれている/含まれていないを発見することができます。Bloom Filterにはまれに偽陽性(含まれていないのに、含まれていると判定する)がありますが、私はこれを自分と全くかかわりのないユーザーが表示されるスパイスのようなものだと考えています。



フロントエンド

今後の展望

- 識別子としてのデジタルアイデンティティと、他者から認識されることで獲得するアイデンティティのうまい併用
- SimHashやEmbeddingを用いた、近いコンテンツほどネットワーク的に近いノードに配置されるindexレイヤーの開発
- プライベートなコミュニケーションの検討

などに取り組み、引き続き新たなWeb世界を模索したいです。